Information page for written examinations at Linköping University

Examination date	2019-08-20
Room (1)	TER1(12)
Time	14-18
Edu. code	TDDC78
Module	TEN1
Edu. code name **Module name**	Programming of Parallel Computers - Methods and Tools (Programmering av parallelldatorer - metoder och verktyg) Written examination (Skriftlig tentamen)
Department	IDA
Number of questions in the examination	8
Teacher responsible/contact person during the exam time	Examinator, kursansvarig, jour: Christoph Kessler (0703-666687)
Contact number during the exam time	+46 703 666687 (C. Kessler)
Visit to the examination room approximately	15:30, afterwards on travel, reachable via 0703-666687
Name and contact details to the course administrator (name + phone nr + mail)	Veronica Kindeland-Gunnarsson 013-285634 veronica.kindeland.gunnarsson@liu.se
Equipment permitted	English dictionary
Other important information	See general instructions on page 1 of the exam.
Number of exams in the bag	

TENTAMEN / EXAM
TDDC78
Programmering av parallelldatorer / Programming of parallel computers
2019-08-20, 14:00–18:00, TER1

Christoph Kessler
Dept. of Computer and Information Science (IDA)
Linköping University

Hjälpmedel / Allowed aids: Engelsk ordbok / dictionary from English to your native language

Examinator: Christoph Kessler

Jourhavande lärare:
Christoph Kessler (IDA), visiting at ca. 15:30, then on travel (reachable via 0703-666687);

Maximalt antal poäng / Max. #points: 40

Betyg / Grading (prel.): The preliminary threshold for passing (grade 3) is at 20p, for grade 4 at 28p, for grade 5 at 34p.
Because of regulations by Linköping University, we do not give ECTS grades. If you need one, please contact the course secretary after the result has been entered in LADOK.

Tentavisning / Exam review session: for the main exam, about 2.5-3 weeks after the exam, to be announced on the course homepage. Afterwards, the exams will be archived in the IDA student expedition. There is no exam review session for re-exams.

General instructions

- Please use a new sheet of paper for each assignment. Order your sheets by assignments, number them, and mark them on top with your exam ID number and the course code.
- You may answer in either English or Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.

1. (7 p.) **Performance tools and analysis**

   (a) (1 p.) Which performance data collection method is required in order to be able to draw a communication statistics diagram (displaying the amount of bytes communicated between every pair of processes)? Justify your answer (technical reasons).

   (b) (2 p.) Describe performance analysis through tracing. (1p)

   What are the advantages and disadvantages of the approach, in comparison to alternative techniques for performance data collection? (1p)

   (c) (1p) Modern tool-suites for performance analysis of parallel programs consist of a collection of several kinds of tools. Give four different kinds of such tools. *(I.e., no concrete tool names, but a short term saying what each kind of tool does.)*

   (d) (2p) Why should the PRAM (Parallel Random Access Machine) model of parallel computing be used *early* in the process of designing and analyzing parallel algorithms?

   And why should it better be replaced by some other cost model in the *later* phases of parallel program design? Give one *example* of such a parallel cost model that could be used for analysis when targeting MPI programs on a cluster system like Tetralith.

   (e) (1p) How is the so-called *peak performance* of a modern cluster-based parallel computer system such as Tetralith calculated? *(derive a commented formula).*

   In particular, which assumptions are made for determining the peak performance?

2. (4 p.) **Parallel program design methodology**

   Foster's design methodology consists of four stages. Name and explain them. Give details about their goals. What are the important properties of the result of each stage? Be thorough!

3. (5 p.) **Parallel computer architecture**

   (a) (1.5p) Many bus-based shared-memory systems use *bus-snooping* for cache coherence. What does that mean, and how does it work? In particular, under what condition can it guarantee sequential consistency?

   (b) (1.5p) What does *hardware multithreading* mean?

   And what is the main difference between a hardware-multithreaded processor and a *multi-core* processor?

   Under what condition (on the computations made by these threads) can hardware multithreading help to improve performance (throughput)?

   (c) (2p) For each of the following requirements, suggest the most appropriate interconnection network topology/topologies (among those discussed in the lecture):

      i. Good fit for an on-chip network of a many-core CPU

      ii. Efficient broadcast at low cost

      iii. Network diameter grows logarithmically with the number of nodes

iv. Network cost linear in the number of nodes, constant node degree, *and* suitable for large-scale weather forecast with a domain decomposition along all three space dimensions

Shortly justify your answers.

4. (7 p.) **OpenMP**

   (a) (1.5p) You are given a working and reasonably well-structured sequential legacy code for some performance-critical numerical simulation, written in C or Fortran. The computational core of the large code base consists of a few dozen `for` loops most of which appear to be parallelizable (such as dataparallel loops) or partly parallelizable (such as reduction loops).

   Your job is to parallelize the code for a shared-memory system to achieve some speedup, and you do not have much time for that task. Moreover, the original author of the code wants to be able to modify it later.

   (i) If you have the choice between Pthreads and OpenMP, which one do you choose for this task, and why? (0.5p)

   (ii) Where do you start and how do you proceed? (Main steps, no details) (1p)

   (b) (2p) Most OpenMP work-sharing constructs, such as the parallel for/do loops, allow to specify an optional `nowait` clause. For example,

```
#pragma omp for nowait
for (i=0; i<N; i++)
  { ... }
#pragma omp for
for (j=0; j<M; j++)
  { ... }
```

   (i) Describe the effect of the `nowait` clause on the computation of the executing threads. (0.5p)

   (ii) Why can it be beneficial for performance to use `nowait`? (0.5p)

   (iii) Formulate a sufficient condition (dependence-based argument) on the two loops in the example above for when it is safe to use `nowait` in the first of the two loops. (1p)

   (c) (2.5p) What kind of loop-based computations can benefit from using the `reduction` clause in OpenMP?

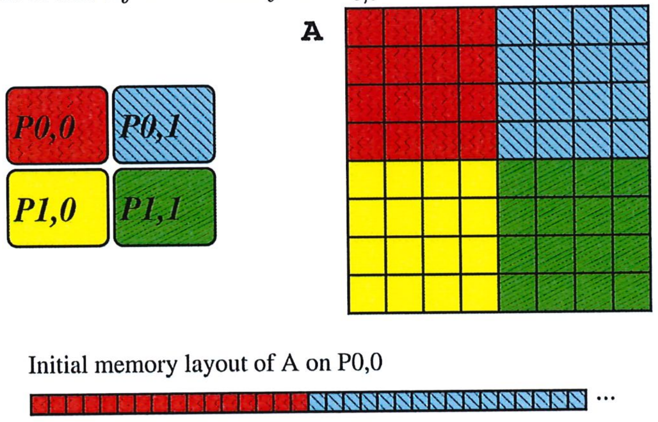
   Give also one example loop (OpenMP code without `reduction`), explain how the generated code for the loop will work instead when using `reduction` (be thorough), and give 2 main reasons why using `reduction` is likely to improve performance.

   (d) What is the main difference between a *PGAS programming language* and a shared memory programming language like OpenMP? (1p)

5. (9 p.) **Parallel Basic Linear Algebra**

(a) (1p) Why is it very important for a supercomputing center to have efficient implementations of BLAS installed?

(b) **MPI and Matrix-Vector Product**:
You are given a distributed-memory parallel computer with $p = r^2$ nodes (for simplicity, using one process per node) organized as a $r \times r$ square grid of nodes (processes); see the figure on the left for the case $p = 4$.
You are also given a $n \times n$ square matrix A and a n-element vector b stored initially on process $P_{0,0}$. Assume that each $n/r \times n/r$ square partition is initially already stored consecutively in memory on $P_{0,0}$.



i. Which feature does MPI offer in order to simplify the indexing of MPI processes in communication in such a 2D processor grid? Explain shortly what it does and how it could be used here. (1p)

ii. Write a *MPI program* (exact syntax is not required, MPI pseudocode is fine, explain your code) that computes the *matrix-vector product* $x = Ab$ in parallel (distributing the work across all p processors), where the matrix A, initially residing entirely on process $P_{0,0}$ only, should be distributed in a *2D-block* manner as shown for $p = 4$ in the right figure, i.e., each process will own a $(n/r) \times (n/r)$ square block of elements once distributed; assume for simplicity that r divides n.
For this given distribution of A, choose a suitable data distribution of the vectors b and x for the distributed computation; motivate your choice.
At the end, the result vector x should be available on process $P_{0,0}$.
Use suitable BLAS calls for the sequential computations performed in each MPI process.
Use suitable collective MPI communication operations wherever applicable.
Explain your code.
Draw also a processor-time diagram that shows the ordering of work phases and communication. (5p)

iii. Assume for simplicity that the p nodes are directly connected to each other with a dedicated direct link for each pair of nodes. Analyze the parallel execution time of the program above using the Delay cost model, i.e., sending and receiving a block of k elements takes time $t_{comm}(k) = \alpha k + \beta$ for constants α, $\beta > 0$. If you need to make further assumptions, state them carefully. (2p)

6. (2 p.) **Parallel Solving of Linear Equation Systems**

   When considering the second step in Foster's design method, we discussed as an example the update schemes of iterative solver methods for linear equation systems, including *Jacobi*-style and *Gauss-Seidel*-style update schemes, over the matrix elements in each iteration step. For each of them, explain its dependence structure between element updates and its maximum degree of parallelism within one iteration of the algorithm.

7. (1 p.) **Transformation and Parallelization of Sequential Loops**

   (a) Is the following C loop parallelizable (in this form)?

   ```
   for (i=0; i<N; i++) {
      t[i] = t[i] + sin(3.1415 * t[i-1] + a[i]);
   }
   ```

   Explain why or why not (dependence-based argument). (1p)

8. (5 p.) **MPI**

   (a) (1.5p) Today's HPC clusters have multi-core nodes. How can, in general, MPI and OpenMP parallelization be suitably combined to leverage hybrid (MPI+OpenMP) parallelism in the same application?

   (b) (1.5p) How does the *nonblocking* (also called *incomplete*) send routine `MPI_Isend` differ from the ordinary blocking `MPI_Send`?
   Under what condition is it safe to replace a `MPI_Send` call by `MPI_Isend`?

   (c) (2 p.) *One-sided communication in MPI*

      i. Explain the principle of one-sided communication in MPI. (1.5p)
      Hint: You might want to illustrate your answer with a pseudocode example and/or an annotated picture.

      ii. Why is one-sided communication considered being "closer" to the shared-memory programming model than ordinary two-sided message passing? (0.5p)